**1. Uso Corpóreo de Comentários**

Os comentários são úteis, mas podem ser mais bem estruturados ou colocados em blocos antes de trechos do código. Exemplos de justificativas melhor senha.

**AÇÕES À MELHORAR APÓS: Vou continuar de onde parei para garantir que todas as análises importantes e sugestões sejam fornecidas:**

**Continuando sugestões e melhorias no design do schema do Mongoose**

1. **Uso de Tipos do Mongoose**
   * Algumas propriedades como loja\_id, codigo, fornecedor, etc., aceitam String. Dependendo do caso, pode ser mais eficiente ou semântico usar ObjectId (para IDs, associados a outra coleção no MongoDB) ou Number em vez de String. Por exemplo:

javascriptCopiar

loja\_id: { type: ObjectId, ref: 'outraColecao', required: false}

1. **Adicionando Validações e Restrições**
   * Embora você tenha usado required: false para várias propriedades (o que, na verdade, é o padrão no Mongoose), é uma boa prática adicionar validações como maxlength, min, enum, etc., para evitar inconsistências nos dados.
   * Exemplo:

javascriptCopiar

cidade: {type: String, required: false, maxlength: 100},

qte: {type: Number, min: 0},

referência: {type: String, required: true},

precocusto: {type: Number, min: 0},

1. **Correção de Date.now () no createdAt**
   * O Date.now () está sendo invocado diretamente na configuração do default. No Mongoose, você deve passar a referência da função, em vez de invocá-la diretamente. Corrija para:

javascriptCopiar

createAt: {

type: Date,

default: Date.now

}

1. **Controlar Atualização de updatedAt Automaticamente**
   * O campo updateAt pode ser atualizado automaticamente usando a opção de timestamps do Mongoose. Em vez de gerenciá-lo manualmente, você pode ativar essa opção ao criar o esquema:

javascriptCopiar

const sample\_Mconstrucao = new Schema (

{

// ... campos aqui

},

{

timestamps: {createdAt: 'createAt', updatedAt: 'updateAt' }

}

);

1. **Estruturas de Dados Complexas (localização)**
   * A forma como localização está estruturada pode ser confusa ao consultar ou atualizar documentos:

javascriptCopiar

localização: [

{

segmento: String,

setor: [

{

nameSetor: String,

secao: [{nameSecao: String}]

}

]

}

]

* + O Mongoose não suporta atualizações em arrays aninhados de forma tão eficiente quanto estruturas mais simples. Considere separar essas estruturas em subdocumentos ou coleções relacionadas:

javascriptCopiar

segmento: {type: ObjectId, ref: 'Segmentos’}

1. **Padronizando as Nomenclaturas**
   * Atualmente o código misturou padrões de nomenclatura. Por exemplo:
     + createAt deveria ser created\_at ou createdAt (lembre-se que o camelCase é mais comum em JS).
     + Campos como marcaloja poderiam ser marca\_loja (snake\_case).
2. **Adicionar Índices**
   * Adicionar índices melhora o desempenho para consultas em campos frequentemente usados em filtros e ordenações. No seu schema, isso faria sentido para campos como:
     + loja\_id
     + codigo
     + cidade
     + descricao

Exemplo de índice:

javascriptCopiar

sample\_Mconstrucao.index({loja\_id: 1});

sample\_Mconstrucao.index({cidade: 1, bairro: 1}, {unique: false});

1. **Melhor Tratamento de Valores Numéricos (Decimais)**
   * Para campos financeiros como precocusto, precovista e precoprazo, usar o tipo Decimal128 (do MongoDB) é uma alternativa interessante para maior precisão decimal:

javascriptCopiar

precocusto: {type: mongoose.Types.Decimal128, required: false},

1. **Boa Prática: Configurar Opções Globais do Schema**
   * Sugestão: configure algumas opções globais ao inicializar o schema para consistência, como a remoção de \_\_v e o uso de toJSON/toObject para deixar as respostas padronizadas:

javascriptCopiar

const sample\_Mconstrucao = new Schema (

{

// campos do schema

},

{

timestamps: {createdAt: 'createAt', updatedAt: 'updateAt’},

versionKey: false, // remove \_\_v

toJSON: {virtuals: true},

toObject: {virtuals: true}

}

);

1. **Validação Adicional de Dados**
   * Caso o código lide com grandes volumes de operações ou importações, pode ser útil aplicar validações customizadas (via validate) para garantir a integridade.
   * Exemplo:

javascriptCopiar

precostusto: {

type: Number,

validate: {

validator: function (v) {

return v >= 0;

},

message: 'O preço de custo deve ser um valor positivo!'

}

}

**Código Melhorado**

Abaixo está o schema já incorporando as sugestões mencionadas:

javascriptCopiar

const mongoose = require('mongoose');

const mongoosePaginate = require('mongoose-paginate-v2');

const Schema = mongoose.Schema;

const sample\_Mconstrucao = new Schema (

{

loja\_id: {type: mongoose.Types.ObjectId, ref: 'Loja', required: false},

marcaloja: {type: String, required: false},

cidade: {type: String, required: false, maxlength: 100},

bairro: {type: String, required: false},

codigo: {type: String, required: false},

fornecedor: {type: String, required: false},

descricao: {type: String, required: false},

complete: {type: String, required: false},

referência: {type: String, required: true},

qte: {type: Number, required: false, min: 0},

qte\_negativa: {type: Number, required: false, min: 0},

qte\_reservada: {type: Number, required: false, min: 0},

e\_max: {type: Number, required: false},

e\_min: {type: Number, required: false },

precocusto: { type: mongoose.Types.Decimal128, required: false},

precovista: { type: mongoose.Types.Decimal128, required: false},

precoprazo: { type: mongoose.Types.Decimal128, required: false},

artigo: {type: String, required: false},

marcaItem: {type: String, required: false},

page: {type: String, required: false},

pageposicao: {type: Number, required: false},

pageurl: {type: String, required: false},

pageok: {type: String, required: false},

ponto: {type: Number, required: false},

figure\_mini: {type: String, required: false},

figure\_média: {type: String, required: false},

localizacao: [

{

segmento: {type: String, required: false},

setor: [

{

nameSetor: {type: String, required: false},

secao: [{nameSecao: {type: String, required: false} }]

}

]

}

]

},

{

timestamps: {createdAt: 'createAt', updatedAt: 'updateAt’},

versionKey: false,

toJSON: {virtuals: true},

toObject: {virtuals: true }

}

);

sample\_Mconstrucao.plugin(mongoosePaginate);

mongoose.model('sample\_Mconstrucao', sample\_Mconstrucao);

**Resumo das Alterações**

1. Ajustado o Date.now () no createdAt.
2. Aplicada a opção de timestamps para gerenciar criação e atualização.
3. Adicionados índices para melhorar o desempenho.
4. Melhoradas validações de valores numéricos e strings.
5. Usado Decimal128 para campos financeiros.
6. Simplificado e padronizado nomes de campos.

Essas alterações garantirão melhor desempenho, consistência e precisão ao manipular grandes volumes de dados no MongoDB. 😊

agora quero disponibilizar em uma lista numa page Html. Trabalho com node e handlebars. Também quero colocar na lista um botaõ de alterar e outro delete